# Application 2: Real-Time Data Processing System for Weather Monitoring with Rollups and Aggregates

## Codebase:

```
from flask import Flask, jsonify, render_template
import sqlite3

app = Flask(__name__)

# Connect to the SQLite database
def get_db_connection():
    conn = sqlite3.connect('weather_data.db')
    conn.row_factory = sqlite3.Row
    return conn

# API route to fetch current weather data
@app.route('/api/weather')
def get_weather():
    conn = get_db_connection()
    cursor = conn.execute('SELECT * FROM weather_summary ORDER BY date DESC LIMIT 6')
    weather_data = cursor.fetchall()
    conn.close()

    weather_summary = []
    for row in weather_data:
        weather_summary.append({
            'city': row['city'],
            'date': row['date'],
```

```
            'avg_temp': row['avg_temp'],
            'max_temp': row['max_temp'],
            'min_temp': row['min_temp'],
            'dominant_weather': row['dominant_weather']
        })
    return jsonify(weather_summary)

# Serve frontend
@app.route('/')
def index():
    return render_template('index.html')

if __name__ == '__main__':
    app.run(debug=True)
```

**HTML (index.html)**:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Real-Time Weather Monitoring</title>
    <link rel="stylesheet" href="/static/styles.css">
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
<body>
    <div class="container">
        <h1>Real-Time Weather Monitoring</h1>
```

```
    <!-- Section for displaying current weather -->
    <div id="current-weather">
        <h2>Current Weather</h2>
        <div class="cities">
            <!-- Weather data will be dynamically populated here -->
        </div>
    </div>

    <!-- Section for daily summaries -->
    <div id="daily-summary">
        <h2>Daily Weather Summary</h2>
        <div class="summary-cards">
            <!-- Weather summary will be populated here -->
        </div>
    </div>

    <!-- Section for historical trends -->
    <div id="historical-trends">
        <h2>Historical Weather Trends</h2>
        <canvas id="tempChart"></canvas>
    </div>

    <!-- Section for alerts -->
    <div id="alerts">
        <h2>Weather Alerts</h2>
        <div class="alert-box">
            <!-- Alerts will be displayed here -->
        </div>
    </div>
</div>
```

```
    <script src="/static/app.js"></script>
</body>
</html>
```

**CSS (styles.css)**:

```
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    color: #333;
}

.container {
    width: 80%;
    margin: 0 auto;
    padding: 20px;
}

h1, h2 {
    text-align: center;
}

.cities, .summary-cards {
    display: flex;
    justify-content: space-around;
    margin: 20px 0;
}

.city-card, .summary-card {
    background-color: #fff;
    border: 1px solid #ddd;
```

```
    padding: 15px;
    border-radius: 5px;
    width: 30%;
    text-align: center;
}

.alert-box {
    text-align: center;
    padding: 20px;
    border: 2px solid red;
    border-radius: 5px;
    background-color: #ffd5d5;
    color: red;
    font-weight: bold;
    margin: 20px 0;
}
```

**JavaScript (app.js)**:

```
document.addEventListener("DOMContentLoaded", () => {
    const currentWeatherContainer = document.querySelector('.cities');
    const summaryContainer = document.querySelector('.summary-cards');
    const alertsContainer = document.querySelector('.alert-box');
    const tempChartCtx = document.getElementById('tempChart').getContext('2d');

    // Fetch weather data from the backend API
    fetch('/api/weather')
        .then(response => response.json())
        .then(data => {
            // Display current weather
            data.forEach(cityData => {
```

```
    const cityCard = document.createElement('div');
    cityCard.classList.add('city-card');
    cityCard.innerHTML = `
        <h3>${cityData.city}</h3>
        <p>Avg Temp: ${cityData.avg_temp.toFixed(2)}°C</p>
        <p>Max Temp: ${cityData.max_temp.toFixed(2)}°C</p>
        <p>Min Temp: ${cityData.min_temp.toFixed(2)}°C</p>
        <p>Condition: ${cityData.dominant_weather}</p>
    `;
    currentWeatherContainer.appendChild(cityCard);

    // Display daily summary
    const summaryCard = document.createElement('div');
    summaryCard.classList.add('summary-card');
    summaryCard.innerHTML = `
        <h4>${cityData.city} - ${cityData.date}</h4>
        <p>Avg Temp: ${cityData.avg_temp.toFixed(2)}°C</p>
        <p>Max Temp: ${cityData.max_temp.toFixed(2)}°C</p>
        <p>Min Temp: ${cityData.min_temp.toFixed(2)}°C</p>
        <p>Dominant Weather: ${cityData.dominant_weather}</p>
    `;
    summaryContainer.appendChild(summaryCard);

    // Show an alert if a condition is met
    if (cityData.max_temp > 35) {
        alertsContainer.innerHTML += `
            <p>ALERT: ${cityData.city} has exceeded 35°C!</p>
        `;
    }
});
```

```
    // Create a chart for historical trends
    const tempChart = new Chart(tempChartCtx, {
        type: 'line',
        data: {
            labels: data.map(cityData => cityData.date),
            datasets: data.map(cityData => ({
                label: cityData.city,
                data: [cityData.avg_temp],
                fill: false,
                borderColor: 'rgb(75, 192, 192)',
                tension: 0.1
            }))
        },
        options: {
            scales: {
                x: {
                    title: {
                        display: true,
                        text: 'Date'
                    }
                },
                y: {
                    title: {
                        display: true,
                        text: 'Temperature (°C)'
                    }
                }
            }
        }
    });
})
```

```
    .catch(error => console.error('Error fetching weather data:', error));
}
```

**API Integration (OpenWeatherMap):**

```
import requests
import time

API_KEY = 'your_openweather_api_key'  # Replace with your API key
BASE_URL = 'http://api.openweathermap.org/data/2.5/weather'

cities = ["Delhi", "Mumbai", "Chennai", "Bangalore", "Kolkata", "Hyderabad"]

def get_weather_data(city):
    url = f"{BASE_URL}?q={city}&appid={API_KEY}"
    response = requests.get(url)
    return response.json()

def process_weather_data():
    weather_data = {}
    for city in cities:
        data = get_weather_data(city)
        if data.get('cod') == 200:  # Check if API call is successful
            main = data['weather'][0]['main']
            temp = data['main']['temp'] - 273.15  # Convert from Kelvin to Celsius
            feels_like = data['main']['feels_like'] - 273.15
            timestamp = data['dt']
            weather_data[city] = {
                'main': main,
                'temp': temp,
                'feels_like': feels_like,
                'timestamp': timestamp
```

```
            }
    return weather_data

# Simulate fetching data every 5 minutes
if __name__ == "__main__":
    while True:
        data = process_weather_data()
        print(data)
        time.sleep(300)  # Sleep for 5 minutes
```

Data Processing and Aggregation:

```
from datetime import datetime
from collections import defaultdict

# Aggregation structure
daily_summaries = defaultdict(lambda: {
    'temps': [],
    'feels_like': [],
    'weather_conditions': []
})

def aggregate_data(weather_data):
    for city, details in weather_data.items():
        day = datetime.utcfromtimestamp(details['timestamp']).strftime('%Y-%m-%d')
        daily_summaries[city]['temps'].append(details['temp'])
        daily_summaries[city]['feels_like'].append(details['feels_like'])
        daily_summaries[city]['weather_conditions'].append(details['main'])

def calculate_daily_summary(city):
    if daily_summaries[city]['temps']:
```

```
        avg_temp = sum(daily_summaries[city]['temps']) / len(daily_summaries[city]['temps'])
        max_temp = max(daily_summaries[city]['temps'])
        min_temp = min(daily_summaries[city]['temps'])
        dominant_weather = max(set(daily_summaries[city]['weather_conditions']),
key=daily_summaries[city]['weather_conditions'].count)
        return {
            'avg_temp': avg_temp,
            'max_temp': max_temp,
            'min_temp': min_temp,
            'dominant_weather': dominant_weather
        }
    return None

# Example usage after data aggregation
# Call aggregate_data periodically and use calculate_daily_summary at the end of the day
```

**Database Setup:**

```
import sqlite3

# Connect to SQLite database
conn = sqlite3.connect('weather_data.db')
cursor = conn.cursor()

# Create a table to store daily summaries
cursor.execute('''
CREATE TABLE IF NOT EXISTS weather_summary (
    city TEXT,
    date TEXT,
    avg_temp REAL,
```

```
    max_temp REAL,
    min_temp REAL,
    dominant_weather TEXT
)
''')

# Store the summary in the database
def store_summary(city, summary, date):
    cursor.execute('''
    INSERT INTO weather_summary (city, date, avg_temp, max_temp, min_temp, dominant_weather)
    VALUES (?, ?, ?, ?, ?, ?)
    ''', (city, date, summary['avg_temp'], summary['max_temp'], summary['min_temp'],
summary['dominant_weather']))
    conn.commit()

# Example of storing data
date = '2024-10-21'
summary = calculate_daily_summary('Delhi')
store_summary('Delhi', summary, date)
```

**Alerting System:**

```
ALERT_THRESHOLD = 35  # Example threshold

def check_alerts(weather_data):
    for city, details in weather_data.items():
        if details['temp'] > ALERT_THRESHOLD:
            print(f"ALERT: {city} temperature exceeded {ALERT_THRESHOLD}°C at {details['temp']}°C")

# Add this to your data processing loop
```

**Visualization:**

```
import matplotlib.pyplot as plt


def plot_summary(city, summary):
    dates = [entry[1] for entry in summary]
    temps = [entry[2] for entry in summary]  # Assuming the 2nd column is avg_temp


    plt.plot(dates, temps, label=city)
    plt.xlabel('Date')
    plt.ylabel('Average Temperature (°C)')
    plt.title(f"Temperature Trends for {city}")
    plt.legend()
    plt.show()


# You can query data from the database and call plot_summary.
```